Parallel Adaptive Solvers in Compressible PETSc-FUN3D Simulations*

S. Bhowmick,[a] D. Kaushik,[b] L. McInnes,[b] B. Norris,[b] P. Raghavan [c]

[a]Department of Applied Physics and Applied Mathematics, Columbia University,
200 S.W. Mudd Building, 500 W. 120th Street, New York, NY 10027,
E-mail: *sb2423@columbia.edu*

[b]Mathematics and Computer Science Division, Argonne National Laboratory,
9700 South Cass Avenue, Argonne, IL 60439-4844,
E-mail: *[kaushik,mcinnes,norris]@mcs.anl.gov*

[c]Department of Computer Science and Engineering, The Pennsylvania State University,
343K IST Building, University Park, PA 16802-6106, E-mail: *raghavan@cse.psu.edu*

Abstract. We consider parallel, three-dimensional transonic Euler flow using the PETSc-FUN3D application, which employs pseudo-transient Newton-Krylov methods. Solving a large, sparse linear system at each nonlinear iteration dominates the overall simulation time for this fully implicit strategy. This paper presents a polyalgorithmic technique for adaptively selecting the linear solver method to match the numeric properties of the linear systems as they evolve during the course of the nonlinear iterations. Our approach combines more robust, but more costly, methods when needed in particularly challenging phases of solution, with cheaper, though less powerful, methods in other phases. We demonstrate that this adaptive, polyalgorithmic approach leads to improvements in overall simulation time, is easily parallelized, and is scalable in the context of this large-scale computational fluid dynamics application.

1. INTRODUCTION

Many time-dependent and nonlinear computational fluid dynamics (CFD) applications involve the parallel solution of large-scale, sparse linear systems. Typically, application developers select a particular algorithm to solve a given linear system and keep this algorithm fixed throughout the simulation. However, it is difficult to select *a priori* the most effective algorithm for a given application. Moreover, for long-running applications in which the numerical properties of the linear systems change as the simulation progresses, a single algorithm may not be best throughout the entire simulation. This situation has motivated us to develop an adaptive, polyalgorithmic approach for linear solvers, which this paper discusses in the context of parallel, three-dimensional transonic Euler flow using PETSc-FUN3D [2]. This application employs pseudo-transient Newton-Krylov methods for a fully implicit solution. We

*This work was supported in part by the National Science Foundation through grants ACI-0102537, CCR-0075792, CCF-0352334, CCF-0444345, ECS-0102345 and EIA-022191 and by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract W-31-109-ENG-38 and DE-AC03-76SF00098.

present a technique for adaptively selecting the linear solver methods to match the numeric properties of the linearized Newton systems as they evolve during the course of the nonlinear iterations. Our approach combines more robust, but more costly, methods when needed in particularly challenging phases of the solution, with faster, though less powerful, methods in other phases. Our previous work focused on sequence-based adaptive heuristics in an uniprocessor environment [4, 5, 9]. In this paper, we extend our research to solve a more complicated parallel application, where we demonstrate that the adaptive polyalgorithmic approach can be easily parallelized, is scalable, and can lead to improvements in overall simulation time.

The remainder of this paper is organized as follows. Section 2 introduces our motivating application. Section 3 explains our approach to adaptive solvers, and Section 4 presents some parallel experimental results. Section 5 discusses conclusions and directions of future work.

2. PARALLEL COMPRESSIBLE FLOW EXAMPLE

FUN3D is an unstructured mesh code originally developed by W. K. Anderson of the NASA Langley Research Center [1] for solving the compressible and incompressible Navier-Stokes equations. This code uses a finite volume discretization with a variable order Roe scheme on a tetrahedral, vertex-centered mesh. Anderson et al. subsequently developed the parallel variant PETSc-FUN3D [2], which incorporates MeTiS [7] for mesh partitioning and the PETSc library [3] for the preconditioned Newton-Krylov family of implicit solution schemes.

This paper focuses on solving the unsteady compressible three-dimensional Euler equations

$$\frac{\partial u}{\partial t} + \frac{1}{V} \oint_{\Omega} \left(\overrightarrow{F} \cdot \hat{n} \right) d\Omega = 0, \tag{1}$$

where

$$u = \begin{bmatrix} \rho \\ \rho u_x \\ \rho u_y \\ \rho u_z \\ E \end{bmatrix}, \qquad \overrightarrow{F} \cdot \hat{n} = \begin{bmatrix} \rho U \\ \rho U u_x + \hat{n_x} p \\ \rho U u_y + \hat{n_x} p \\ \rho U u_z + \hat{n_z} p \\ (E+p) U \end{bmatrix},$$

$$U = \hat{n_x} u_x + \hat{n_y} u_y + \hat{n_z} u_z, \qquad p = (\gamma - 1) \left[E - \rho \frac{(u_x^2 + u_y^2 + u_z^2)}{2} \right].$$

Here ρ, E, and U represent the density, energy, and velocity in the direction of the outward normal to a cell face, respectively. The pressure field p is determined by the equation of state for a perfect gas (given above). Also the cell volume (V) is enclosed by the cell boundary (Ω).

We solve the nonlinear system in Equation (1) with pseudo-timestepping [8] to advance towards an assumed steady state. Using backward Euler time discretization, Equation (1) becomes

$$\frac{V}{\Delta t^{\ell}} (u^{\ell} - u^{\ell-1}) + f(u^{\ell}) = 0, \tag{2}$$

where $\Delta t^{\ell} \rightarrow \infty$ as $\ell \rightarrow \infty$, u is a vector of unknowns representing the state of the system, and $f(u)$ is a vector-valued function of residuals of the governing equations, which satisfies $f(u) = 0$ in the steady state.

This code employs Roe's flux-difference splitting to discretize the convective terms. Initially Equation (2) is discretized by using a first-order scheme. When the nonlinear residual sinks

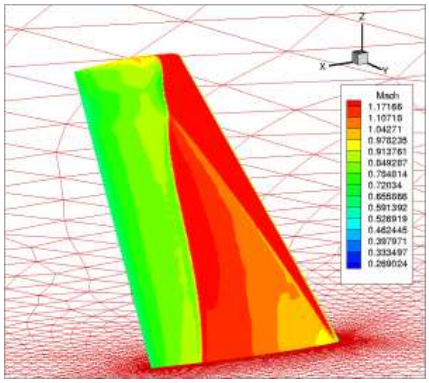

Figure 1. Mach contours on the ONERA M6 wing at freestream Mach number = 0.839.

below a given threshold, a second-order discretization is applied. The timestep is advanced toward infinity by a power-law variation of the switched evolution/relaxation (SER) heuristic of Mulder & Van Leer [10]. To be specific, within each of the first-order and second-order phases of computation we adjust the timestep according to the CFL number,

$$N_{CFL}^{\ell} = N_{CFL}^{0} \left(\frac{\|f(u^0)\|}{\|f(u^{\ell-1})\|} \right)^{\sigma},$$

where σ is normally unity but is damped to 0.75 for robustness in cases in which shocks are expected to appear.

At each timestep we apply a single inexact Newton iteration (see, e.g., [11]) to Equation (2) through the two-step sequence of (approximately) solving the Newton correction equation

$$(\frac{V}{\Delta t^{\ell}} I + f'(u^{\ell-1}))\ \delta u^{\ell} = -f(u^{\ell-1}), \tag{3}$$

where I is the identity matrix, and then updating the iterate via $u^{\ell} = u^{\ell-1} + \delta u^{\ell}$. We employ matrix-free Newton-Krylov methods (see, e.g., [6]), with which we compute the action of the Jacobian on a vector v by directional differencing of the form $f'(u)v \approx \frac{f(u+hv)-f(u)}{h}$, where h is a differencing parameter. We use a first-order analytic discretization to compute the corresponding preconditioning matrix.

We explore the standard aerodynamics test case of transonic flow over an ONERA M6 wing using the frequently studied parameter combination of a freestream Mach number of 0.839 with an angle of attack of 3.06^o. The robustness of solution strategies is particularly important for this model because of the so-called λ-shock that develops on the upper wing surface, as depicted in Figure 1. As mentioned earlier, the PDEs are discretized by using a first-order scheme; but once the shock position has settled down, a second-order discretization is applied. This change in discretization affects the nature of the resulting linear systems. The time for the complete simulation is dominated by the time to solve the linear systems generated at each nonlinear iteration, where this phase typically requires around 77 percent of the overall time. Moreover, changes in the numerical characteristics of the linear systems reflect the changing nature of the simulation. For example, the use of pseudo-transient continuation generates linear systems

that become progressively more difficult to solve as the simulation advances. This situation is discussed further in Section 4; in particular, see the right-hand graph of Figure 2.

3. ADAPTIVE SOLVERS

Applications where the properties of the linear systems differ significantly throughout the simulation, such as PETSc-FUN3D's [2] modeling of compressible Euler flow, are an ideal case for exploring adaptive solvers. Our goal is to improve overall performance by combining more robust (but more costly) methods when needed in particularly challenging phases of solution with faster (though less powerful) methods in other phases. Adaptive solvers are designed with the goal of reducing overall execution time by dynamically selecting the most appropriate method to match the needs of the current linear system.

Adaptive solvers can be defined by the heuristic employed for method selection. The efficiency of an adaptive heuristic depends on how appropriately it determines *switching points*, or the iterations at which to change linear solvers. Adaptive heuristics monitor changes in *indicators* to detect switching points. We have observed that a combination of several indicators generally leads to better results. Some common examples of indicators are linear (nonlinear) solution time, convergence rate, change in nonlinear residual norm, etc.

In this paper we employ sequence-based adaptive heuristics, which rely on a predetermined sequence of linear solvers and then "switch up" to a more robust but more costly method or "switch down" to a cheaper but less powerful method as needed during the simulation. In this class of heuristics, only three methods are compared when making a given switching point decision – the current method and the methods directly preceding and succeeding it in the sequence. Adaptive heuristics are nonsequence-based when *all* the methods in the available set are compared. This class of adaptive methods requires more time for method selection than does the sequence-based class but has greater flexibility. Sequence-based methods are used in simulations where linear systems tend to become progressively difficult or easier; nonsequence-based strategies are used in applications where a monotonic pattern is missing. As discussed in Sections 2 and 4, the PETSc-FUN3D simulation falls in the former category.

We employed the following two indicators to construct the adaptive, polyalgorithmic solver:

- *Nonlinear residual norm:* The pseudo-transient continuation depends on the CFL number [8], which, as explained in Section 2 and shown in the left-hand side of Figure 2, increases as the nonlinear residual norm decreases. As the CFL number increases, the corresponding Newton correction equations (3) become more difficult to solve. Thus, in this case the nonlinear residual norm is a good indicator of the level of difficulty of solving its corresponding Newton correction equation: the lower the residual norm, the more difficult the linear system is likely to be. Based on trial runs of the application, we divided the simulation into four sections: (a) $||f(u)|| \geq 10^{-2}$, (b) $10^{-4} \leq ||f(u)|| < 10^{-2}$, (c) $10^{-10} \leq ||f(u)|| < 10^{-4}$, and (d) $||f(u)|| < 10^{-10}$. Whenever the simulation crosses from one section to another, the adaptive method switches up or down accordingly.

- *Average time per nonlinear iteration:* Our second indicator provides a rough estimate of the strength of the linear solver. The higher the value of the indicator, the more likely the solver is to effectively solve difficult linear systems. The base solvers are arranged in increasing order of their corresponding indicator values.

The parallelization of the adaptive scheme was straightforward; we invoked the linear solvers as determined by the heuristic, without redistributing the parallel data. No changes were needed to either the compressible Euler code or the base parallel solvers in PETSc [3] to accommodate this adaptive approach.

4. EXPERIMENTAL RESULTS

To perform numerical experiments on the compressible Euler application introduced in Section 2, we used the Jazz cluster at Argonne National Laboratory, which has a Myrinet 2000 network and 2.4 GHz Pentium Xeon processors with 1-2 GB of RAM. Our experiments focus on a problem instance designated as 1Grid (with 357,900 vertices and 2.4 million edges), which generates a Jacobian matrix of rank approximately 1.8×10^6 with 1.3×10^8 nonzeros. The large problem size makes it imperative to use a multiprocessor architecture. We ran the simulations on 4, 8, 16, and 32 processors using various Krylov methods and various subdomain solvers for a block Jacobi preconditioner with one block per processor. The relative linear convergence tolerance was 10^{-3}, and the maximum number of iterations for any linear solve was 30. The left-hand side of Figure 2 shows how the CFL number increases as the nonlinear residual norm decreases for the pseudo-transient Newton-Krylov algorithm; this situation was discussed in more detail in Section 2. The right-hand side of Figure 2 shows the time per nonlinear iteration for various solvers on four processors.

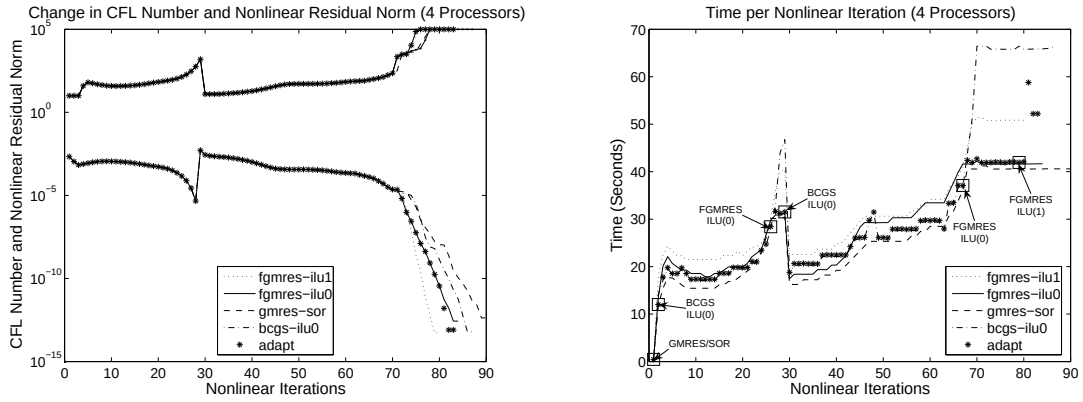

Figure 2. *Left:* Convergence rate (lower plot) and CFL number (upper plot) for the base and adaptive solvers on 4 processors. *Right:* Time per nonlinear iteration for the base and adaptive solvers on 4 processors.The labeled square markers indicate when linear solvers changed in the adaptive algorithm.

We employed four linear solvers: (1) GMRES with a block Jacobi (BJ) preconditioner that uses SOR as a subdomain solver, designated as GMRES-SOR; (2) bi-conjugate gradient squared (BCGS) with a BJ preconditioner that uses no-fill incomplete factorization (ILU(0)) as a subdomain solver, called BCGS-ILU0; (3) flexible GMRES (FGMRES) with a BJ preconditioner that uses ILU(0) as a subdomain solver, designated as FGMRES-ILU0; and (4) FGMRES with a BJ

preconditioner that uses ILU(1) as a subdomain solver, called FGMRES-ILU1. We considered these as traditional base methods that remain fixed throughout the nonlinear solution process, and we also combined them in an adaptive scheme as introduced in Section 3. We ordered these methods for use in the adaptive solver as (1), (2), (3), (4), according to the average time taken per nonlinear iteration in the first-order discretization phase of the simulation, which can serve as a rough estimate of the strength of the various linear solvers for this application. The graphs in Figure 3 show the switching points among these methods in the adaptive polyalgorithmic approach. The simulation starts with method (1), then switches to method (2) at the next iteration. The switch to method (3) occurs at iteration 25. The discretization then shifts to second order at iteration 28, and the initial linear systems become easier to solve. The adaptive method therefore switches down to method (2). From this point onward, the linear systems become progressively more difficult to solve as the CFL number increases; the adaptive method switches up to method (3) in iteration 66 and method (4) in iteration 79. In the right-hand graph of Figure 2, this last change is accompanied by an increase in the time taken for the succeeding nonlinear iteration. This increased time is devoted to setting up the new preconditioner, which in this case changes the block Jacobi subdomain solver from ILU(0) to ILU(1) and consequently requires more time for the factorization phase.

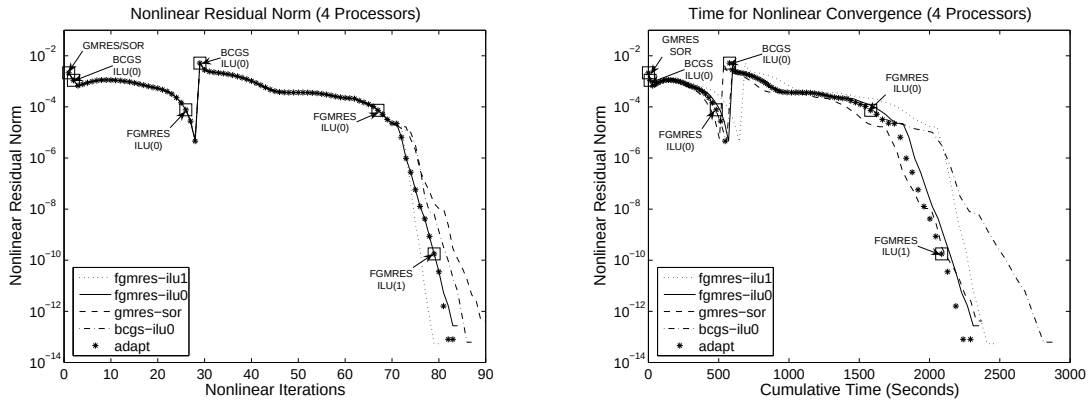

Figure 3. Performance of base and adaptive methods on 4 processors. *Left:* Residual norm vs. nonlinear iteration number. *Right:* Residual norm vs. cumulative simulation time. The labeled square markers indicate when linear solvers changed in the adaptive algorithm. The first solution method was GMRES-SOR, and switching occurred at iterations 1, 25, 28, 66, and 79.

The execution time of the adaptive polyalgorithmic scheme is 3% better than the fastest base method (FGMRES-ILU0) and 20% better than the slowest one (BCGS-ILU0). We also observe that the final nonlinear residual norm obtained by using the adaptive method is comparable to that obtained from the best base method ($O(10^{-14})$). In addition, the number of linear iterations required by the adaptive method for the overall simulation is smaller than that needed by any of the base methods. The overhead for switching among base methods in the adaptive scheme is minimal: approximately $0.02\% - .06\%$ of the total execution time.

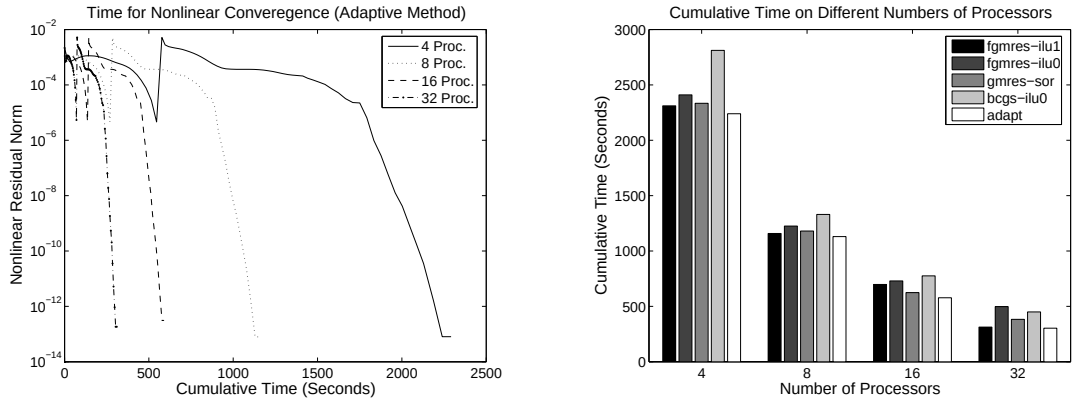

Figure 4. Performance of base and adaptive algorithms on 4, 8, 16, and 32 processors.

Figure 4 shows the performance of the adaptive method on 4, 8, 16, and 32 processors. As the number of processors increases, the simulation requires less time to converge. Although the best base method varies with the number of processors (FGMRES-ILU1 for 4, 8, and 32 processors, to GMRES-SOR for 16 processors), the adaptive method is always the one requiring the least time. The improvement of the adaptive scheme compared to the best base method varies from 2% to 7%. Moreover, as it is impossible to know *a priori* what particular linear solution algorithm will be fastest for a long-running nonlinear simulation, the adaptive, polyalgorithmic approach adjusts linear solvers according to the levels of difficulty encountered throughout a given run.

We consider the speedups of the adaptive and base solvers in Figure 5. Since this large-scale problem size requires a minimum of 4 processors, we use T_1 as the best estimate of the time for the full simulation on one processor using the fastest base method. We set T_1 to four times the time required with FGMRES-ILU1. We calculate the speedup $S = \frac{T_1}{T_p}$, where T_p is the observed time on p processors. The results show that the speedup of the adaptive method is almost ideal and as good as or better than any of the base methods.

5. CONCLUSIONS AND FUTURE WORK

In this paper we presented an adaptive, polyalgorithmic approach that dynamically selects a method to solve the linearized systems that arise in the PETSc-FUN3D application's modeling of compressible Euler flow using a pseudo-transient Newton-Krylov method. This approach reduced overall execution time by using cheaper though less powerful linear solvers for relatively easy linear systems and then switching to more robust but more costly methods for more difficult linear systems. Our results demonstrate that adaptive solvers can be implemented easily in a multiprocessor environment and are scalable. We are now investigating adaptive solvers in additional problem domains and considering more adaptive approaches, including a polynomial heuristic where the trends of the indicators can be estimated by fitting a function to the known data points. We also are combining adaptive heuristics with high-performance component infrastructure for performance monitoring and analysis, as described in [12, 13].

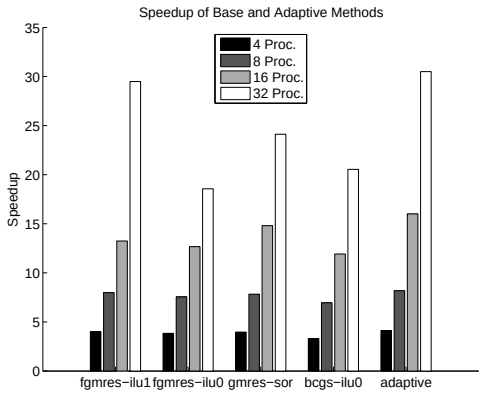

Figure 5. Scalability of the base and adaptive algorithms on 4, 8, 16, and 32 processors.

REFERENCES

1. W. K. Anderson and D. Bonhaus. An implicit upwind algorithm for computing turbulent flows on unstructured grids. *Computers and Fluids*, 23(1):1–21, 1994.
2. W. K. Anderson, W. D. Gropp, D. K. Kaushik D. E. Keyes, and B. F. Smith. Achieving high sustained performance in an unstructured mesh CFD application. In *Proceedings of Supercomputing 1999*. IEEE Computer Society, 1999. Gordon Bell Prize Award Paper in Special Category.
3. S. Balay, K. Buschelman, W. Gropp, D. Kaushik, M. Knepley, L. McInnes, Barry F. Smith, and H. Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 2.2.1, Argonne National Laboratory, 2004. `http://www.mcs.anl.gov/petsc`.
4. S. Bhowmick. Multimethod Solvers, Algorithms, Applications and Software, 2004. Ph.D. Thesis, Department of Computer Science and Engineering, The Pennsylvania State University.
5. S. Bhowmick, L. C. McInnes, B. Norris, and P. Raghavan. The role of multi-method linear solvers in PDE-based simulations. *Lecture Notes in Computer Science, Computational Science and its Applications-ICCSA 2003*, 2667:828–839, 2003.
6. P. N. Brown and Y. Saad. Hybrid Krylov methods for nonlinear systems of equations. *SIAM Journal on Scientific and Statistical Computing*, 11:450–481, 1990.
7. G. Karypis and V. Kumar. A fast and high quality scheme for partitioning irregular graphs. *SIAM Journal of Scientific Computing*, 20:359–392, 1999.
8. C. T. Kelley and D. E. Keyes. Convergence analysis of pseudo-transient continuation. *SIAM Journal on Numerical Analysis*, 35:508–523, 1998.
9. L. McInnes, B. Norris, S. Bhowmick, and P. Raghavan. Adaptive sparse linear solvers for implicit CFD using Newton-Krylov algorithms. *Proceedings of the Second MIT Conference on Computational Fluid and Solid Mechanics*, June 17-20,2003.
10. W. Mulder and B. Van Leer. Experiments with implicit upwind methods for the Euler equations. *Journal of Computational Physics*, 59:232–246, 1985.
11. J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer-Verlag, 1999.
12. B. Norris, L. McInnes, and I. Veljkovic. Computational quality of service in parallel CFD. Argonne National Laboratory preprint ANL/MCS-P1283-0805, submitted to *Proc. of the 17th International Conference on Parallel CFD*, Aug 2005.
13. B. Norris and I. Veljkovic. Performance monitoring and analysis components in adaptive PDE-based simulations. Argonne National Laboratory preprint ANL/MCS-P1221-0105, Jan 2005.